Search Results for: **[extensible <and> macro<and> language]**
Found **2,767** of **114,152 searched.**

Warning: Maximum result set of 200 exceeded. Consider refining.

Search within Results

> Advanced Search

> Search Help/Tips

Sort by: Title Publication Publication Date Score

Results 1 - 20 of 200 short listing

Prev Page 1 2 3 4 5 6 7 8 9 10 Next Page

1 Maya: multiple-dispatch syntax extension in Java 100%
Jason Baker , Wilson C. Hsieh
ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2002 Conference on Programming language design and implementation May 2002
Volume 37 Issue 5
We have designed and implemented Maya, a version of Java that allows programmers to extend and reinterpret its syntax. Maya generalizes macro systems by treating grammar productions as generic functions, and semantic actions on productions as multimethods on the corresponding generic functions. Programmers can write new generic functions (i.e., grammar productions) and new multimethods (i.e., semantic actions), through which they can extend the grammar of the language and change the semantics of ...

2 The Java syntactic extender (JSE) 99%
Jonthan Bachrach , Keith Playford
ACM SIGPLAN Notices , Proceedings of the OOPSLA '01 conference on Object Oriented Programming Systems Languages and Applications October 2001
Volume 36 Issue 11
The ability to extend a language with new syntactic forms is a powerful tool. A sufficiently flexible macro system allows programmers to build from a common base towards a language designed specifically for their problem domain. However, macro facilities that are integrated, capable, and at the same time simple enough to be widely used have been limited to the Lisp family of languages to date. In this paper we introduce a macro facility, called the Java Syntactic Extender (JSE), with the superio ...

3 A brief look at extension programming before and now 99%

Liisa Räihä

ACM SIGPLAN Notices February 1995

Volume 30 Issue 2

We try to bind together some old and some new: what is an extension. In addition, we give a short analysis of extension facilities in three language systems with slightly different theoretical basis. We compare Ada 9x, Oberon and Macro Language, with additional comments to e.g., C++.

4 Hygienic macro expansion 99%

Eugene Kohlbecker , Daniel P. Friedman , Matthias Felleisen , Bruce Duba

Proceedings of the 1986 ACM conference on LISP and functional programming August 1986

5 Multics Emacs (Prose and Cons): A commercial text-processing system in Lisp 97%

Bernard S. Greenberg

Proceedings of the 1980 ACM conference on LISP and functional programming August 1980

This paper addresses the choice of Lisp as the implementation language, and its consequences, including some of the implementation issues. The detailed history of Multics Emacs, its system-level design considerations, and its impact on Multics and its user community are discussed in [Greenberg]. One of the immediate and profound consequences of this choice has been to assert Lisp's adequacy, indeed, superiority, as a full-fledged systems and applications programming language. Multics Emacs ...

6 Composable and compilable macros:: you want it when? 97%

Matthew Flatt

ACM SIGPLAN Notices , Proceedings of the seventh ACM SIGPLAN international conference on Functional programming September 2002

Volume 37 Issue 9

Many macro systems, especially for Lisp and Scheme, allow macro transformers to perform general computation. Moreover, the language for implementing compile-time macro transformers is usually the same as the language for implementing run-time functions. As a side effect of this sharing, implementations tend to allow the mingling of compile-time values and run-time values, as well as values from separate compilations. Such mingling breaks programming tools that must parse code without executing i ...

7 A history of the SNOBOL programming languages 97%

Ralph E. Griswold

The first ACM SIGPLAN conference on History of programming languages January 1978

Development of the SNOBOL language began in 1962. It was followed by SNOBOL2, SNOBOL3, and SNOBOL4. Except for SNOBOL2 and SNOBOL3 (which were closely related), the others differ substantially and hence are more properly considered separate languages than versions of one language. In this paper historical emphasis is placed on the original language, SNOBOL, although important aspects of the subsequent languages are

covered.

8 Systems semantics: principles, applications, and implementation 96%
Ray Boute
ACM Transactions on Programming Languages and Systems (TOPLAS) January 1988
Volume 10 Issue 1
Systems semantics extends the denotational semantics of programming languages to a semantics for the description of arbitrary systems, including objects that are not computations in any sense. By defining different meaning functions, the same formal description may be used to denote different system properties, such as structure, behavior, component cost, and performance aspects (e.g., timing). The definition of these semantic functions also provides guidance in language design, ...

9 Using meta-level compilation to check FLASH protocol code 96%
Andy Chou , Benjamin Chelf , Dawson Engler , Mark Heinrich
Proceedings of the ninth international conference on Architectural support for programming languages and operating systems November 2000
Volume 34 , 28 Issue 5 , 5
Building systems such as OS kernels and embedded software is difficult. An important source of this difficulty is the numerous rules they must obey: interrupts cannot be disabled for ~too long," global variables must be protected by locks, user pointers passed to OS code must be checked for safety before use, etc. A single violation can crash the system, yet typically these invariants are unchecked, existing only on paper or in the implementor's mind.This paper is a case study in how system impl ...

10 Macros as multi-stage computations: type-safe, generative, binding macros in MacroML 96%
Steven E. Ganz , Amr Sabry , Walid Taha
ACM SIGPLAN Notices , Proceedings of the sixth ACM SIGPLAN international conference on Functional programming October 2001
Volume 36 Issue 10
With few exceptions, macros have traditionally been viewed as operations on syntax trees or even on plain strings. This view makes macros seem ad hoc, and is at odds with two desirable features of contemporary typed functional languages: static typing and static scoping. At a deeper level, there is a need for a simple, usable semantics for macros. This paper argues that these problems can be addressed by formally viewing macros as multi-stage computations. This view eliminates the need for fresh ...

11 Translator writing systems 96%
Jerome Feldman , David Gries
Communications of the ACM February 1968
Volume 11 Issue 2
A critical review of recent efforts to automate the writing of translators of programming languages is presented. The formal study of syntax and its application to translator writing are discussed in Section II. Various approaches to automating the postsyntactic (semantic) aspects of translator writing are discussed in Section III, and several related topics in Section IV.

12 Using meta-level compilation to check FLASH protocol code 96%

Andy Chou , Benjamin Chelf , Dawson Engler , Mark Heinrich
ACM SIGPLAN Notices November 2000
Volume 35 Issue 11
Building systems such as OS kernels and embedded software is difficult. An important source of this difficulty is the numerous rules they must obey: interrupts cannot be disabled for "too long," global variables must be protected by locks, user pointers passed to OS code must be checked for safety before use, etc. A single violation can crash the system, yet typically these invariants are unchecked, existing only on paper or in the implementor's mind.This paper is a case study in how system impl ...

13 VCODE: a retargetable, extensible, very fast dynamic code generation system 95%
Dawson R. Engler
ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1996 conference on Programming language design and implementation May 1996
Volume 31 Issue 5
Dynamic code generation is the creation of executable code at runtime. Such "on-the-fly" code generation is a powerful technique, enabling applications to use runtime information to improve performance by up to an order of magnitude [4, 8,20, 22, 23].Unfortunately, previous general-purpose dynamic code generation systems have been either inefficient or non-portable. We present VCODE, a retargetable, extensible, very fast dynamic code generation system. An important feature of VCODE is that it ge ...

14 Modern languages and Microsoft's component object model 95%
David N. Gray , John Hotchkiss , Seth LaForge , Andrew Shalit , Toby Weinberg
Communications of the ACM May 1998
Volume 41 Issue 5

15 Pointcuts and advice in higher-order languages 94%
David B. Tucker , Shriram Krishnamurthi
Proceedings of the 2nd international conference on Aspect-oriented software development March 2003
Aspect-oriented software design will need to support languages with first-class and higher-order procedures, such as Python, Perl, ML and Scheme. These language features present both challenges and benefits for aspects. On the one hand, they force the designer to carefully address issues of scope that do not arise in first-order languages. On the other hand, these distinctions of scope make it possible to define a much richer variety of policies than first-order aspect languages permit.In this p ...

16 How to write system-specific, static checkers in metal 94%
Benjamin Chelf , Dawson Engler , Seth Hallem
ACM SIGSOFT Software Engineering Notes , Proceedings of the 2002 ACM SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering November 2002
Volume 28 Issue 1

17 SLX: pyramid power 94%
James O. Henriksen

Proceedings of the 31st conference on Winter simulation: Simulation---a bridge to the future - Volume 1 December 1999

18 The architecture of Montana: an open and extensible programming environment with an incremental C++ compiler 94%

Michael Karasick

ACM SIGSOFT Software Engineering Notes , Proceedings of the 6th ACM SIGSOFT international symposium on Foundations of software engineering November 1998 Volume 23 Issue 6

Montana is an open, extensible integrated programming environment for C++ that supports incremental compilation and linking, a persistent code cache called a CodeStore, and a set of programming interfaces to the CodeStore for tool writers. CodeStore serves as a central source of information for compiling, browsing, and debugging. CodeStore contains information about both the static and dynamic structure of the compiled program. This information spans files, macros, declarations, function bodies, ...

19 A module system for scheme 94%

Pavel Curtis , James Rauen

Proceedings of the 1990 ACM conference on LISP and functional programming May 1990

This paper presents a module system designed for large-scale programming in Scheme. The module system separates specifications of objects from their implementations, permitting the separate development, compilation, and testing of modules. The module system also includes a robust macro facility. We discuss our design goals, the design of the module system, implementation issues, and our future plans.

20 BASIC Zgrass—a sophisticated graphics language for the Bally Home Library Computer 94%

Tom DeFanti , Jay Fenton , Nola Donato

Proceedings of the 5th annual conference on Computer graphics and interactive techniques August 1978

Home computer users are just now discovering computer graphics. Modest extensions to BASIC allow plotting but not much more. The Bally Home Library Computer, however, has hardware to aid implementation of video games. Custom integrated circuits working on a 160×102 pixel (2 bits per pixel) color television screen allow certain forms of animation in real time. To give this power to the user, BASIC Zgrass has been designed and implemented. It is an extension of BASIC that allows paralle ...

Citation

Conference on Object Oriented Programming Systems Languages and Applications >archive
Conference proceedings on Object-oriented programming systems, languages and applications >toc
1989 , New Orleans, Louisiana, United States

Programming with explicit metaclasses in Smalltalk-80

> Also published in ...

Authors
J.-P. Briot
P. Cointe

Sponsor
SIGPLAN : ACM Special Interest Group on Programming Languages

Publisher
ACM Press New York, NY, USA

Pages: 419 - 431 Series-Proceeding-Article
Year of Publication: 1989
ISSN:0362-1340

doi http://doi.acm.org/10.1145/74877.74921 (Use this link to Bookmark this page)

> full text > abstract > references > citings > index terms > peer to peer

> Discuss > Similar > Review this Article Save to Binder

> BibTex Format

↑ FULL TEXT: Access Rules

pdf 1.33 MB

↑ ABSTRACT

This paper discusses the introduction of explicit metaclasses á la ObjVlisp into the Smalltalk-80 language. The rigidity of Smalltalk metaclass architecture motivated this work. We decided to implement the ObjVlisp model into the standard Smalltalk-80 system. The resulting combination defines the Classtalk platform. This platform provides a full-size environment to experiment with class-oriented programming by combining implicit metaclasses á la Smalltalk and explicit metaclasses á la ObjVlisp. Obviously, these experiments are not limited to the Smalltalk world and will be useful to understand and

practice the metaclass concept advocated by modern object-oriented languages such as ObjVlisp and CLOS.

↑ REFERENCES

Note: OCR errors may be found in this Reference List extracted from the full text article. ACM has opted to expose the complete List rather than only correct and linked references.

Attardi&al89 G. Attardi, C. Bonini, M.-R. Boscotrecase, T. Flagella and M. Gaspari, Metalevel Programming in CLOS, ECOOP'89, Cambridge University Press, July 1989.

Bobrow&Kiczales88 Daniel G. Bobrow , Gregor Kiczales, The common Lisp object system metaobject kernel: a status report, Proceedings of the 1988 ACM conference on LISP and functional programming, p.309-315, July 25-27, 1988, Snowbird, Utah, United States

Borning&OShea87 Alan Borning , Tim O'Shea, Deltatalk: an empirically and aesthetically motivated simplification of the Smalltalk-80 language, European conference on object-oriented programming on ECOOP '87, p.1-10, October 1987, Paris, France

Briot&Cointe87 J.-P. Briot and P. Cointe, A Uniform Model for Object-Oriented Languages Using The Class Abstraction, IJCAr87, Vol. 1, pages 40-43, August 1987.

Cointe87 Pierre Cointe, Metaclasses are first class: The ObjVlisp Model, Conference proceedings on Object-oriented programming systems, languages and applications, p.156-162, October 04-08, 1987, Orlando, Florida, United States

Cointe&Graube88 P. Cointe and N. Graube, Programming with Metaclasses in CLOS, First CLOS Users and Implementors Workshop, Xerox Parc, Palo Alto CA, USA, pages 23-29, October 1988.

Cointe88 P. Cointe, A Tutorial Introduction to Metaclass Architecturesas Provided by ClassOriented Languages, International Conference on Fifth Generation Computer Systems (FGCS'88), Vol. 2, pages 592-608, Icot, Tokyo, Japan, November-December 1988.

Goldberg&Robson83 Adele Goldberg , David Robson, Smalltalk-80: the language and its implementation, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 1983

Graube89 N. Graube, Metaclass compatibility, ACM SIGPLAN Notices, v.24 n.10, p.305-315, Oct. 1989

Ingalls&Borning82 D.H.H. ingalls and A.H. Borning, Multiple inheritance in Smalltalk-80, Proceedings of the National Conference on Artificial Intelligence, pages 234-237, USA, August 1982.

Malenfant&al89 Malenfant, G. Lapalme and J. Vaucher, ObjVProlog: Metaclasses in Logic, ECOOP'89, Cambridge University Press, July 1989.

Ungar&Smith87 David Ungar , Randall B. Smith, Self: The power of simplicity, Conference proceedings on Object-oriented programming systems, languages and applications, p.227-242, October 04-08, 1987, Orlando, Florida, United States

Wolinski89 F. Wolinski, Le Systdme MV2C~ Modglisation et Ggndration dTnterfaces Homme-Machine, Report 89/38, Laforia, Universit6 Pierre et Marie Curie, Paris, April 1989.

↑ CITINGS *6*

Hafedh Mili , Francois Pachet , Ilham Benyahia , Fred Eddy, Metamodeling in OO: OOPSLA'95 workshop summary, ACM SIGPLAN OOPS Messenger, v.6 n.4, p.105-110, Oct 1, 1995

Noury M. N. Bouraqadi-Saâdani , Thomas Ledoux , Fred Rivard, Safe metaclass programming, ACM SIGPLAN Notices, v.33 n.10, p.84-96, Oct. 1998

Mira Mezini, Maintaining the consistency of class libraries during their evolution, ACM SIGPLAN Notices, v.32 n.10, p.1-21, Oct. 1997

Ira R. Forman , Michael H. Conner , Scott H. Danforth , Larry K. Raper, Release-to-release binary compatibility in SOM, ACM SIGPLAN Notices, v.30 n.10, p.426-438, Oct. 17, 1995

Ira R. Forman , Scott Danforth , Hari Madduri, Composition of before/after metaclasses in SOM, ACM SIGPLAN Notices, v.29 n.10, p.427-439, Oct. 1994

Andreas Paepcke, PCLOS: stress testing CLOS experiencing the metaobject protocol, ACM SIGPLAN Notices, v.25 n.10, p.194-211, Oct. 1990

↑ INDEX TERMS

Primary Classification:
D. Software
↳ **D.1** PROGRAMMING TECHNIQUES

Additional Classification:
D. Software
↳ **D.3** PROGRAMMING LANGUAGES
↳ **D.3.2** Language Classifications

↳ **Nouns:** Smalltalk-80; CLOS
↳ **D.3.3** Language Constructs and Features
↳ **Subjects:** Data types and structures

General Terms:
Design, Languages

↑ Peer to Peer - Readers of this Article have also read:

- We Talk to Everybody
  Linux Journal 2000, 74es

Marjorie Richardson , Jason Schumaker , David Penn

- Editorial pointers
  Communications of the ACM 44, 9
  Diane Crawford

- News track
  Communications of the ACM 44, 9
  Robert Fox

- Forum
  Communications of the ACM 44, 9
  Diane Crawford

- At the Forge
  Linux Journal 1998, 52es
  Reuven M. Lerner

↑ This Article has also been published in:

- ACM SIGPLAN Notices
  Volume 24 , Issue 10 (October 1989)

Search Results

Search Results for: **[uncompiled <and> extensible <and> macro <and> language<AND>((smalltalk <and> extensible <and> macro <and> language<AND>((smalltalk<AND>((extensible <and> macro<and> language))))))]**
Found **1** of **114,152 searched.**

Search within Results

Sort by: Title Publication Publication Date Score Binder

Results 1 - 1 of 1 short listing

1 The implementation of procedurally reflective languages 77%
Jim des Rivières , Brian Cantwell Smith
Proceedings of the 1984 ACM Symposium on LISP and functional programming August 1984
In a procedurally reflective programming language, all programs are executed not through the agency of a primitive and inaccessible interpreter, but rather by the explicit running of a program that represents that interpreter. In the corresponding virtual machine, therefore, there are an infinite number of levels at which programs are processed, all simultaneously active. It is therefore a substantial question to show whether, and why, a reflective language is computationally tractable. We ...

Results 1 - 1 of 1 short listing